```
INTERFACE LinkedList ;

PROCEDURE Init();

PROCEDURE ToFirst() ;
PROCEDURE ToLast() ;
PROCEDURE Forth() ;
PROCEDURE Back() ;

PROCEDURE IsAfter() : BOOLEAN ;
PROCEDURE IsBefore() : BOOLEAN ;
PROCEDURE Get() : INTEGER ;
PROCEDURE Set(item: INTEGER) ;

PROCEDURE Append(item : INTEGER);
PROCEDURE InsertBefore(item : INTEGER);
PROCEDURE Delete();

END LinkedList.
```

```
MODULE LinkedList ;

TYPE Item = INTEGER ;
     Node = REF RECORD
         key        : Item ;
         prev, next : Node ;
     END ;

VAR  head, tail, current : Node ;


PROCEDURE Init() =
BEGIN
    head := NEW(Node) ;
    tail := NEW(Node) ;
    head^.prev := head ;
    head^.next := tail ;
    tail^.prev := head ;
    tail^.next := tail ;
    current := head ;
END Init ;


PROCEDURE ToFirst() =
BEGIN
    current := head^.next ;
END ToFirst ;


PROCEDURE ToLast() =
BEGIN
    current := tail^.prev ;
END ToLast ;


PROCEDURE IsAfter() : BOOLEAN =
BEGIN
    RETURN current = tail ;
END IsAfter ;


PROCEDURE IsBefore() : BOOLEAN =
BEGIN
    RETURN current = head ;
END IsBefore ;


PROCEDURE Forth() =
BEGIN
    current := current^.next ;
END Forth ;


PROCEDURE Back() =
BEGIN
    current := current^.prev ;
END Back ;


PROCEDURE Get() : INTEGER =
BEGIN
    <* ASSERT NOT (IsBefore() OR IsAfter()) *>

    RETURN current^.key ;
END Get ;
```

LinkedList.m3

```
PROCEDURE Set(item: INTEGER) =
BEGIN
    <* ASSERT NOT (IsBefore() OR IsAfter()) *>

    current^.key := item ;
END Set ;


PROCEDURE Append(item : INTEGER) =
VAR n : Node ;
BEGIN
    n := NEW(Node) ;
    n^.prev := tail^.prev ;
    n^.next := tail ;
    n^.prev^.next := n ;
    n^.next^.prev := n ;
    n^.key := item ;
END Append ;


PROCEDURE InsertBefore(item : INTEGER) =
VAR n : Node ;
BEGIN
    <* ASSERT NOT IsBefore() *>

    n := NEW(Node) ;
    n^.prev := current^.prev ;
    n^.next := current ;
    n^.prev^.next := n ;
    n^.next^.prev := n ;
    n^.key := item ;
END InsertBefore ;


PROCEDURE Delete() =
BEGIN
    <* ASSERT NOT (IsBefore() OR IsAfter()) *>

    current^.next^.prev := current^.prev ;
    current^.prev^.next := current^.next ;
    current := current^.next ;
END Delete ;

BEGIN
END LinkedList .
```

Main.m3

```
MODULE Main ;

IMPORT IO ;
IMPORT LinkedList AS List ;



PROCEDURE WriteList() =
BEGIN
    List.ToFirst() ;
    WHILE NOT List.IsAfter() DO
        IO.PutInt(List.Get()) ;
        IO.Put("\t") ;
        List.Forth() ;
    END ;
    IO.Put("\n") ;
END WriteList ;


BEGIN
    List.Init() ;
    List.Append(1) ;
    List.Append(2) ;
    List.Append(3) ;
    List.Append(4) ;
    List.Append(5) ;

    WriteList() ;

    List.ToFirst() ;
    List.Forth() ;
    List.InsertBefore(6) ;

    WriteList() ;

    List.ToLast() ;
    List.Back() ;
    List.Delete() ;

    WriteList() ;
END Main .
```